Candidate Marks Report

Series : 6 2018

This candidate's script has been assessed using On-Screen Marking. The marks are therefore not shown on the script itself, but are summarised in the table below.

Centre No :		Assessment Code :	H446
Candidate No :		Component Code :	02
Candidate Name :			

Total Marks :	97 / 140

In the table below 'Total Mark' records the mark scored by this candidate.
'Max Mark' records the Maximum Mark available for the question.

Paper:	H446/02
Paper Total:	97 / 140
Question	Total Mark / Max Mark
1a	3 / 3
1b	3 / 3
1ci	5 / 5
1cii	2 / 2
2ai	2 / 2
2aii	1 / 1
2bi	1 / 1
2bii	4 / 4
2ci	1 / 1
2cii	0 / 1
2d	2 / 2
3ai	1 / 1
3aii	0 / 2
3aiii	1 / 2
3b	4 / 7
3c	5 / 9
3d	5 / 6
4a	0 / 6
4b	1 / 2
4c	4 / 9
4d	1 / 2
5a	4 / 4
5bi	2 / 4
5bii	2 / 2
5c	0 / 3
5di	5 / 5
5dii	4 / 7
5diii	4 / 4
6ai	2 / 3
6aii	5 / 5
6aiii	3 / 3

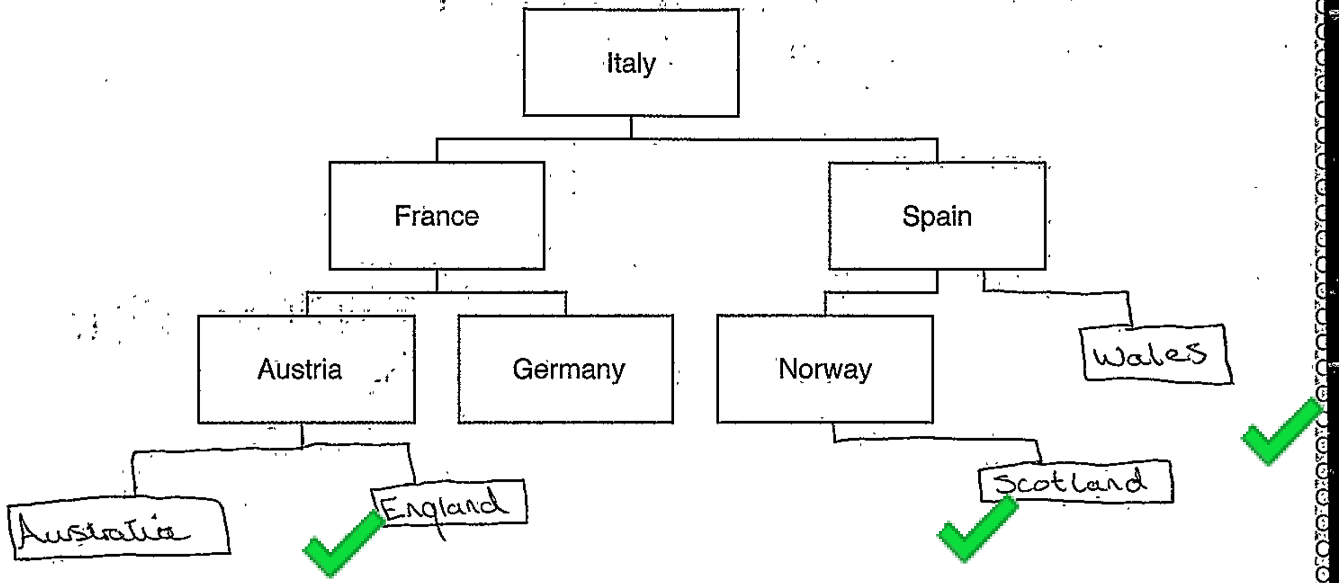
6bi	2 / 2
6bii	0 / 2
6biii	2 / 2
6biv	3 / 6
6bv	1 / 1
6bvi	4 / 5
6bvii	2 / 2
6c	6 / 9

Section A

Answer **all** the questions.

1 A program stores entered data in a binary search tree.

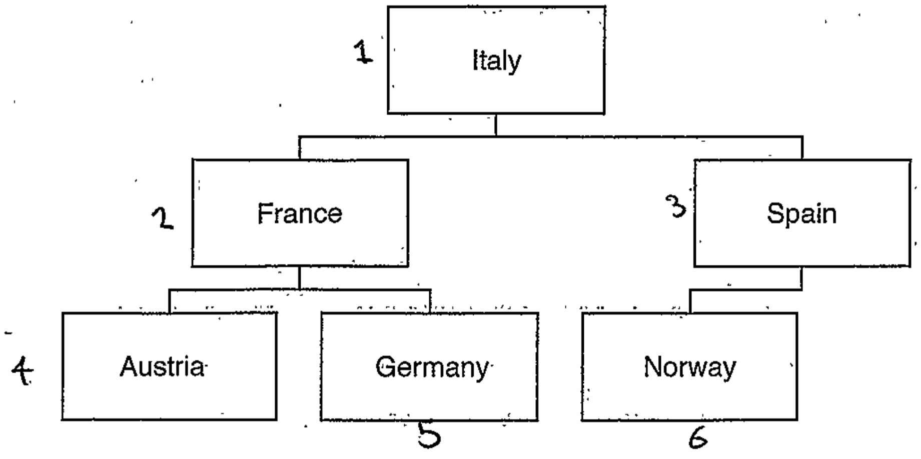
The current contents of the tree are shown:



(a) Complete the diagram to show the contents of the tree after the following data is added:

England, Scotland, Wales, Australia **[3]**

(b) Show the order of the nodes visited in a breadth first traversal on the following tree.



It would first visit Italy, then France, then Spain then Austria then Germany then Norway. It ~~Searches~~ traverses each layer from the ~~f~~ left to the right and once each node at that layer has been visited, it does the same for the next layer. The order of the nodes visited are:
Italy, France, Spain, Austria, Germany, Norway

[3]

(c) A pseudocode algorithm is written to search the tree to determine if the data item "Sweden" is in the tree.

The function `currentNode.left()` returns the node positioned to the left of `currentNode`.

The function `currentNode.right()` returns the node positioned to the right of `currentNode`.

```
function searchForData(currentNode:byVal, searchValue:byVal)
    thisNode = getData(CurrentNode)
    if thisNode == SearchValue then
        return TRUE
    elseif thisNode < searchValue then
        if currentNode.left() != null then
            return (searchForData(currentNode.left(), searchValue))
        else
            return FALSE
        endif
    else
        if CurrentNode.right() != null then
            return (searchForData(currentNode.right(), searchValue))
        else
            return false
        endif
    endif
endfunction
```

(i) Complete the algorithm.

[5]

(ii) The algorithm needs to be used in different scenarios, with a range of different trees.

Identify **two** preconditions needed of a tree for this algorithm to work.

1 Each node cannot have more than ~~one child~~ two children

2 The tree needs to be ordered (The right child has to be bigger and the left child has to be smaller than the current node)

[2]

BLANK PAGE

openRead
Readline
endOfFile
close
openwrite
writeline

PLEASE DO NOT WRITE ON THIS PAGE

2 A company merger is joining five e-commerce retailers under one company, OCRRetail. Each retailer has a different sales system and OCRRetail wants to develop one computer system that can be used by all the retailers.

Mary's software development company has been employed to analyse and design a solution for the company.

(a) **(i)** Two computational methods (techniques used to solve a problem using computational thinking) that Mary will use are problem recognition and decomposition.

State what is meant by problem recognition and decomposition.

Recognition Identifying the problem and it's set of requirements that are needed to produce a solution

Decomposition Breaking down the problem into smaller, more manageable problems, that can each be solved to contribute to the final solution **[2]**

(ii) State **one** additional computational method.

Abstraction **[1]**

(b) Mary plans to use data mining to generate information about OCRRetail's customers. Mary will use this information to benefit the company.

(i) Define the term 'data mining'.

When a large set of data is analysed with the intent to find patterns and relationships within the data **[1]**

(ii) Identify **two** pieces of information that data mining could provide OCRRetail about sales, and state how OCRRetail could make use of this information.

1 A certain type of person (such as a specific age group) purchase a particular product more than any other groups purchase a the product. OCR Retail could then analyse this product and target this group with similar products in the future

2 ~~Certain~~ People who purchase a particular product tend to go on and purchase another specific product. ~~For exam~~ OCR Retail could then do a special offer that combines these two products to encourage the people **[4]** to buy the two

(c) Mary has developed the program and is considering using performance modelling before installing the system.

(i) Define the term 'performance modelling'.

When a solution is presented as a ~~pro~~ abstract solution , especially visa of a graphical interface ✗ **[1]**

(ii) Identify **one** way performance modelling could be used to test the new system.

SEEN

[1]

(d) Mary created the program as a series of sub-programs that can be reused.

Describe **one** benefit of Mary creating reusable program components.

~~The component d~~ It will save her time ✓ and effort as she doesn't have to ~~rewrite~~ recode the same ~~part~~ component again when she wants to use it somewhere else in ✓ the program

[2]

Tested
no rewritten

3 A puzzle has multiple ways of reaching the end solution. Fig. 3 shows a graph that represents all possible routes to the solution. The starting point of the game is represented by A, the solution is represented by J. The other points in the graph are possible intermediary stages.

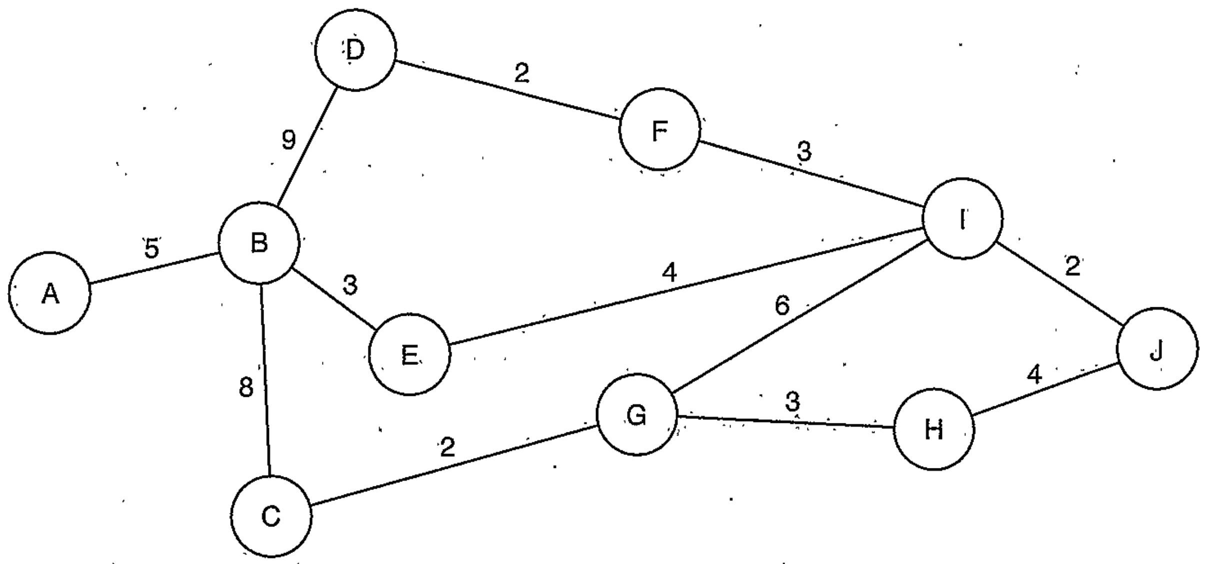


Fig. 3

(a) The graph in Fig. 3 is a visualisation of the problem.

(i) Identify **one** difference between a graph and a tree.

A graph can be cyclic, whereas a tree cannot be cyclic **[1]**

(ii) Explain how the graph is an abstraction of the problem.

The unnecessary information is not shown on the graph. For example, the values on the edges of the graph are shown, however the actual information about how it is achieved to get from one node to another is not shown and hidden in this graph as the distances is only relevent in this representation **[2]**

TV

(iii) Identify **two** advantages of using a visualisation such as the one shown in Fig. 3.

1 The problem will be easier to decompose as the problem is presented in a way that is simple to understand

2

[2]

(b) Demonstrate how Dijkstra's algorithm would find the shortest path to the solution in Fig. 3.

Firstly, each node is put on a priority queue with each value for each node equal to ∞ (infinity). The priority ~~queue~~ queue will be reordered according to the shortest distance node at the front of the queue. A is at the start and all nodes connected directly to A are denoted as partially visited and the node ~~wo~~ with the ~~shortest~~ smallest edge value is moved to the front of the queue with it's value of ∞ changed to it's edge value. A is moved off of the queue before this happens as it has been visited. The same happens with all ~~no~~ nodes directly connected to B ✓ and node E is chosen ✓, with it edge value equal to the value at node B plus the ~~e~~ edge value between B and E, so it is equal to 8. E has one node directly connected so ~~that~~ E is removed from the queue and the distance to node **[7]**
I is the value at E plus the edge value between E and I ✓ which is 8+4=12. ~~I~~ The nodes connected to I are examined and J is the ✓ shortest distance ~~2~~, which is calculated to be 12+2=14. J is the solution so the algorithm stops

(c)* The creator of the puzzle has been told that the A* algorithm is more efficient at finding the shortest path because it uses heuristics.

Compare the performance of Dijkstra's algorithm and the A* search algorithm, making reference to heuristics, to find the shortest path to the problem.

Dijcstras algorithm works by checking every node possible to find the shortest path, but ~~doesn't have a~~ does it this way consistently throughout every problem it deals with, which means it has no idea of how far away it is from the goal node and is only concerned with finding the Shortest path to each node. Dijcstras uses the 'Real cost' function to calculate this, whereas A* uses ~~both~~ the sum of the 'Real cost' function and the 'Heuristic' function to calculate the next node to visit. The use of the Heuristic function is ~~so that~~ to keep the algorithm go on to a path that is always moving towards the goal node. This could be, for example the geographical distance to the goal node if it were finding the path on a GPS. The addition of the Heuristic function to the real cost function makes it better suited for purposes for ~~path~~ GPS or maps to find the shortest path.

The ~~a~~ problem with the A* algorithm is that theoretically, it may not find the actual shortest path to the goal node because, unlike Dijcstras algorithm, it ~~does~~ doesn't check every node so there is no sure way of determining whether the path it is taking is the shortest. ~~D~~ Dijksras algorithm is more suited for mathematical problems as it checks every ~~possible~~ node and so the shortest distance to each node is calculated definitely, whereas A* is more ~~eff~~ efficient for path finding as it makes use of the Heuristic function.

L3

[9]

(d) A computer program version of the puzzle is to be developed. A programmer will use an IDE to debug the program during development.

Describe **three** features of an IDE that help debug the program.

1. Breakpoints ✓ can be used, which ~~is~~ are put at ~~aq~~ a point in the code where the program will stop running when it hits the Breakpoint ✓ This ~~&~~ means lengthy programs don't have to be run ~~ext~~ entirely when the debugger is only testing a smaller part

2. Step Through ✓ can be used to pass a particular part of the program as this particular part may not be finished yet

3. A Watch ✓ can be set on a variable, in which the value of the variable is shown whenever it changes ✓

[6]

4 A recursive function, generate, is shown.

```
function generate(num1:byval)
    if num1 > 10 then
        return 10
    else
        return num1 + (generate(num1 + 1) DIV 2)
    endif
endfunction
```

(a) Trace the algorithm to show the value returned when generate(7) is called. Show each step of your working.

7 is not bigger than 10 so it enters the 'else' block. The line returns the value of 7 plus the value of generate(4 ✗
So the function is recursively called with the value 4.
4 is not bigger than 10 so it enters the else block where it return 4 + generate(2.5) . This will continue forever and result in a stack overflow as it only ever gets smaller and never meets the base case (if num1 > 10) ✗

[6]

(b) The parameter, num1, is passed by value.

Explain why the parameter was passed by value instead of by reference.

If it were passed by reference, the value outside of the function for num1 would change. ✓

[2]

(c)* Parameters can be used to reduce the use of global variables.

Compare the use of parameters to global variables in recursive functions.

Parameters in recursive function allow values to be changed after each call, ~~[illegible]~~ but the value is local to the function so doesn't ~~[illegible]~~ change outside of the function, whereas when it is ~~[illegible]~~ uses global variables, each time it changes, it changes outside the function. If the same variable name is used in another part of the program, it will cause the value to change and may result in things going wrong.

Using parameters in a recursive function is most suited for reusable components as these are ~~[illegible]~~ ones where the variable name is likely used more than once so if it's value is changed throughout the program, it may result in a value changing when it is not meant to.

~~Less~~ Parameters also reduce the amount of global variables, which means less code is ~~[illegible]~~ written and hence reducing the memory of the code.

L2

[9]

(d) A student called Jason writes a recursive algorithm. The recursive algorithm uses more memory than if Jason had written it as an iterative algorithm.

Explain why the recursive algorithm uses more memory than the iterative algorithm.

A recursive algorithm uses a call Stack ✓ which holds the parameters, return address and the local variables. So each time a recursive Subroutine is called, values are added to the Stack which results in an increase in memory usage each time a recursive **[2]**
Subroutine is called. ~~Whereas, an iterative algorithm~~

5 A computer program stores data input on a stack named `dataItems`. The stack has two sub-programs to add and remove data items from the stack. The stack is implemented as a 1D array, `dataArray`.

Sub-program	Description
`push()`	The parameter is added to the top of the stack
`pop()`	The element at the top of the stack is removed

The current contents of `dataItems` are shown:

6
15
100
23

(a) Show the contents of the stack `dataItems` after each line of the following lines of code are run.

```
01  push(13)
02  pop()
03  push(10)
04  push(20)
```

Line 01	Line 02	Line 03	Line 04
			20 ✓
13		10 ✓	10
6 ✓	6	6	6
15	15 ✓	15	15
100	100	100	100
23	23	23	23

[4]

(b) The main program asks a user to push or pop an item from the stack. If the user chooses 'push', the data item is added to the stack. If the user chooses "pop", the next item is removed from the stack, multiplied by 3 and output.

The main program is shown:

```
01  userAnswer = input("Would you like to push or pop an item?")
02  if userAnswer == "push" then
03      push(input("Enter data item"))
04  else
05      print(pop() * 3)
06  endif
```

(i) Before the sub-programs, `push()` and `pop()`, can add or remove items from the stack, a selection statement is used to decide if each action is possible.

Describe the decision that needs to be made in each sub-program and how this impacts the next process.

push() It needs to check if the array is full ✓ ~~is if~~ because ~~th~~ an array is static, ~~it is sorry~~ you cannot push a value if there is no available element ~~that have no value is~~ Therefore, it should check if the stack is full.

pop() If the ~~ro~~ Stack is empty ✓ ~~then~~ pop() cannot return anything as there would be no value to return. Therefore, it should check if the stack is empty.

[4]

(ii) The algorithm does not work when the user enters `"PUSH"` or `"Push"`. The algorithm needs to be changed in order to accept these inputs.

Identify the line number to be changed and state the change that should be made.

Line number 02 ✓

Change if userAnswer == "push" OR userAnswer == "PUSH" ✓ OR userAnswer == "Push" then

[2]

(c) The stack is implemented as a 1D array, `dataArray`.

Describe how a 1D array can be set up and used to push and pop items as a stack.

~~The b~~

~~E~~ The top of the stack can change TV to ~~be~~ point to the ~~so~~ last value in the array and ~~the~~ when a value is popped, this pointer value is popped and then it is changed to point to the next value TV (the value at the end of the array)

[3]

(d) As an array, the data in `dataArray` is sorted and then searched for a specific value.

(i) The data in `dataArray` is sorted into ascending order using an insertion sort.

The current contents of `dataArray` are shown:

100	22	5	36	999	12

Show the steps of an insertion sort on the current contents of the array `dataArray`.

'100' is the first element in the array. The next element is compared with ~~to~~ '100' and it is smaller, so '22' is inserted into position 0 and '100' is moved to position 1. ~~? is compared~~ So far, '100' and '22' have been sorted which means that each other element ('5', '36', '999', '12') will be sorted once inserted into the array. '5' is compared with the current sorted values, which are '100' and '22' and ~~is so~~ inserted into position 0 and '22' is now in position 1 and '100' is in position 2. '36' is compared with '100' and is smaller so is compared with '22' and is bigger so is inserted into position ~~2~~ '3', with the value '100' moved to position '4'. '999' is compared with all of the values currently sorted and is larger so stays in it's current position (position **[5]**
4). '12' is the final value to be sorted and is inserted into position 1 as it is bigger than '5' but smaller than '36'. Because there are no more values to be inserted, the array is sorted.

To Summarise, The final, sorted array is ordered as follows

5	12	22	36	100	999

Note that when I ~~am~~ was referring to positions, I am referring to array notation and not human counting.

(ii) The array `dataArray` can now be searched using a binary search.

Describe the stages of a binary search on an array of size `n`.

The Binary Search uses ~~B~~ a left pointer and a right pointer, which will initially point to the first element and the last element respectively in the array. The midpoint ~~will be~~ is determind to be initially equal to the length of the ~~a~~ array divided by two. Because the array is sorted, the values to the left of the midpoint and the values to the right of midpoint will be smaller and larger than the value of midpoint respectively. If midpoint is smaller than the value being searched, left mark is changed to point to the ~~element~~ next element to the right of the midpoint ~~[illegible]~~ and if the midpoint is greater than the value being searched for, the right mark is changed to point to the ~~a~~ next element to the left of midpoint. The new value of ~~the~~ midpoint is calculated by calculating the sum of the position of leftmark and right mark and dividing this value by 2. Note that if the value of this calculation is not an **[7]**
integer (i.e the value of leftmark + right mark is ~~e~~ an odd number), the value is rounded down. ~~The~~
The same process is carried out on the new values of ~~left pointer,~~ leftmark, right mark and midpoint ~~and~~ until the size ~~has been~~ is equal to 1. ~~T~~ When it is equal to 1, the value being searched has been found.

TV

FT

FT

REP

(iii) The array has 50 items.

The function, `searchItem()`, performs a linear search for a data item.

```
function searchItem(dataItem)
    for count = 0 to 49
        if dataArray[count] == dataItem then
            return(count)
        endif
    next count
    return(-1)
endfunction
```

Rewrite the function using a while loop.

```
function SearchItem(dataItem)
    Count = 0
    while(count < 50)
        if(dataArray[count] == dataItem)then
            ~~return[count]~~
            return(count)
        end if
        Count = count + 1
    end while
    return(-1)
end function
```

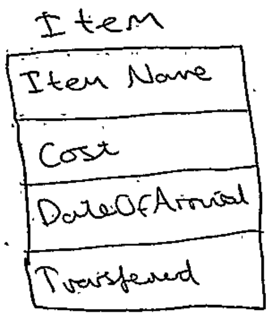
[4]

* 0009657900920 *

Section B

Answer **all** questions.

6 Kamran is writing a program to manipulate the data for a set of items.



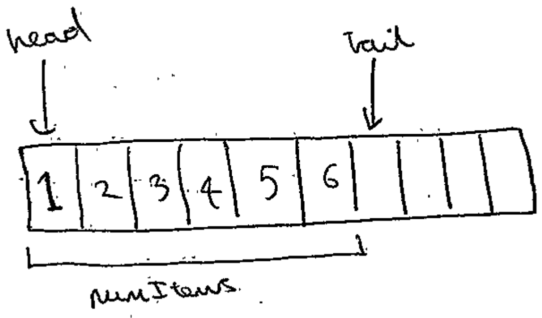
For each item, the program needs to store:

- Item name (e.g. Box)
- Cost (e.g. 22.58)
- Date of arrival (e.g. 1/5/2018)
- Transferred (e.g. true)

The items are added to a queue for processing.

The queue is defined as a class, `itemQueue`.

```
itemQueue
---------------------------
theItems[10] : Items
head : Integer
tail : Integer
numItems : Integer
---------------------------
constructor
enqueuer()
dequeuer()
setnumItems()
getnumItems()
```



The `head` attribute points to the first element in the queue. The `tail` attribute points to the next available space in the queue. The `numItems` attribute states how many items are currently in the queue.

(a) The data about the items can be stored using either a record structure, or as objects of a class.

(i) Explain the similarities and differences between a record and a class.

Both a record and a class can have multiple attributes of different types. ✓

~~Whereas a cla~~

However, a class can have methods ✓ and behaviours, whereas a record cannot

[3]

(ii) Kamran chooses to use a record structure to store the data about the items.

Record structures may be declared using the following syntax:

```
recordStructure recordstructurename
    fieldname : datatype
    ...
endRecordStructure
```

Complete the pseudocode to declare a record called `items`.

```
recordStructure ....items.........✓..........................
    itemName : ....String.......✓.........................
    ~~Cost~~ cost ...✓............................: Currency
    ~~[illegible]~~ dateOfArrival ✓......: Date
    transferred : ...Boolean.......✓.......................
endRecordStructure  ✓
```

[5]

(iii) New records may be created using the following syntax:

```
recordidentifier : recordstructurename
recordidentifier.fieldname = data
...
```

✓ ✓

Write a programming statement to create a new item, using the identifier 'box1', with the item name "Box", the cost 22.58, date of arrival 1/5/2018 and transferred true.

```
box1 : items
box1.cost = 22.58
box1.dateOfArrival = 1/5/2018
box1.itemName = "Box"
box1.transferred = ~~tru~~TRUE
```

[3]

(b) The array, `theItems`, stores the items in the queue. When the tail of the queue exceeds the last element in the array, it adds a new item to the first element if it is vacant.

For example, in the following queue, the next item to be added would be placed at index 0.

Index	0	1	2	3	4	5	6	7	8	9
Element				Data	Data	Data	Data	Data	Data	Data

(i) Define the term 'queue'.

A queue is a First in First out data structure, in which the first data item added to the queue is the first one to be removed from the queue.

[2]

(ii) The attributes in `itemQueue` are all declared as private.

Explain how a private attribute improves the integrity of the data.

The integrity of the data is dependent on it's accuracy, which is why it is private; as if it were public and ~~a~~ an attribute with the same name were to be used, the value would also change in the array. By making it private, encapsulation is used so that all values are accurate and correct **[2]**

TV

(iii) The constructor method creates a new instance of `itemQueue` and sets the `head`, `tail` and `numItems` attributes to 0.

Write an algorithm, using pseudocode or program code, for the constructor including the initialisation for all attributes.

```
public procedure new([illegible] head, tail, numItems)
    ~~this.numItems = numItems~~
    numItems = 0
    tail = 0
    head = 0
end procedure
```

[2]

(iv) The `enqueue` method:

- takes as a parameter the item to insert in the queue item
- checks if the queue is full
- reports an error and returns `false` if the queue is full
- does the following if the queue is not full:
    - o adds the item to the array at the tail position and adjusts the pointer(s)
    - o returns `true`

The attribute `numItems` stores the number of items currently in the queue.

Write an algorithm, using pseudocode or program code, for the `enqueue` method.

```
public function enQueue(item ✓ [illegible] )
    arrayLength = length(theItems)
    if(arrayLength == numItems ✓ )then
        print("The Queue is full")
        return FALSE ✓
    else
        array[tail ✗] = item
        tail = tail+1 ✗
        return True ✗
    end if
end function
```

[6]

(v) Write a programming statement to declare an instance of `itemQueue` called `myItems`.

~~MyItems = new (item~~

myItems = new ItemQueue() **[1]**

(vi) Write a procedure, `insertItems()`, to ask the user to input the data for an item. The item is then added to the queue `myItems`. The user is continually asked to input data items until the queue is full.

```
public procedure insertItems()
    ~~itemName = input("enter an item name:")~~
    ~~cost = input("enter the cost:")~~
    ~~dateOfArrival = input("enter the date of arrival:")~~
    ~~transferred = input("has it been transferred?")~~
    ~~while (length~~
    count = 0
    while(count != length(myItems))
        itemName = input("enter an itemName:")
        cost = input("enter the cost:")
        dateOfArrival = input("enter the date of arrival:")
        transferred = input("has the item been transferred?")
        myItems.enQueue(item)
        count = count + 1
    end while
end procedure
```

[5]

(vii) When the main program ends, the items and the queue no longer exist.

Describe how Kamran could amend the program to make sure the items and queue still exist and are used the next time the program is run.

~~The root~~ The data stored in the queue and the items can be written to a file ✓ each time they are created and when ~~they~~ the program is next used, the file can be read from and the data will therefore be retrievable. The file will have to be saved ✓ after each update (each time it is written to) **[2]**

(c)* Kamran wants to expand the program to allow it to handle up to 100,000,000 items and to allow him to search for data about items. Kamran is worried that the increase in the number of items will cause a decrease in the performance of the program. He decides to investigate the benefits of caching and concurrent processing.

Evaluate the use of caching and concurrent processing in this scenario and make a recommendation to Kamran.

Caching is a technique used in which program instructions and data are temporarily stored locally ~~[illegible]~~ so that it can be retrieved quickly. This is a useful technique to use when particular data is likely to be required again shortly as it reduces the time it takes to access the data. ~~He~~ Kamran could use caching be storing programming instructions that are often used when searching for data items so that these instructions don't have to be continually ~~loaded~~ ~~fro~~ accessed from main memory.

Concurrent processing is when multiple tasks are carried out by the CPU at the same time (using interleaving) allowing for multiple tasks to be able to be run at the same time. For the large data set Kamran is working with, concurrent processing will allow for the multiple tasks that are required to handle 100,000,000 data items ~~[illegible]~~

to be executed quickly. This will increase the peformance as it will allow multiple tasks to be completed at the same time. Caching would improve peformance as it will reduce the time taken for the CPU to access the data ~~as if it~~ as if commonly used data and instructions are continually fetched from main memory, it would produce a bottleneck in the CPU peformance. So caching ultimately will reduce this and allow the CPU to perform more tasks per second.

Kamran should make use of both of these techniques, especially caching and should store the instructions commonly used in the program locally to the CPU.

L2

[9]

END OF QUESTION PAPER

PLEASE DO NOT WRITE ON THIS PAGE

* 0009657900928 *